Inventor's Explanation of a Real World Example of Application of the Present Invention

<Reference Explanation>

I would like to give following explanations for a reference purpose by taking a web site in which the present invention is embodied, aotual real number data and a variable program as examples.

There is a web site in which the present invention is embodied and it is CyberParts™ (Registered trademark of FUJITSU Ltd.) provided by FUJITSU Ltd. This is an Original Equipment Manufacturing (OEM) product of Technote Co., Ltd.
(http://cyberparts.fenics.or.jp/)
I would like to explain the structure of drawing data (real number data and variable program) which is used in the web site as follows:

Explanation of the data structure of CyberParts

The data structure of CyberParts consists of real number data files and variable program files, and all of them take TXT file form. Real number data is read in from real number data files, and operated according to the variable program files. Then, the vector data of a drawing for a plan is created. The good point of this data structure is that in case where there is one form part and it has a difference in many sizes, the vector data of the drawing can be automatically created by parametric method which becomes possible by storing parameter data which corresponds to each variable to real number data. It can be said that this is the best way to store mechanical parts data in a sense of data size owing to the fact that it is parametric data and is also the best way to deliver data on the Web because it is TEXT data.

This data structure is exactly the same structure as "JIS Kikai Parts" soled by Design Automation Inc., which is an OEM product of Technote Co., Ltd. (The above is bundled to the CAD system, "CADPAC" of Design Automation Inc. Please refer the following address)
http://www.dai.com/da-net/products/cadpac/cadpac_jis.shtml
JIS Machine Parts was started to be sold in February 1998 and was publicly known before the patent application of the present invention was made. The present invention is the one which is organized to make parametric drawings possible to be used on the Internet by using the data structure of JIS Machine Parts.

1. CyberParts, Embodiment of Parts Searching Screen

Fig 1 is an embodiment of parts searching screen of CyberParts. It is constituted of "Page Column", "Maker Column", "First Parts List", "Second Parts List", "Third Parts List", and "Fourth Parts List". If the user chose machine element in the Page Column, parts concerning machine elements are displayed in the following parts lists. In the same manner, if the user chose JIS Machine Parts, peaces of information concerning JIS standardized parts are displayed in the following parts list. If the user chose screw parts in First Parts List, a parts list concerning screw parts is displayed in Second Parts List. If the user chose bolts in Second Parts List, a parts list concerning bolts is displayed in Third Parts List. If the user chose hexagon head bolts in Third Parts List, a real number data file of hexagon head bolts is opened in Fourth Parts List and parts type number which is stored in real number data file is displayed.

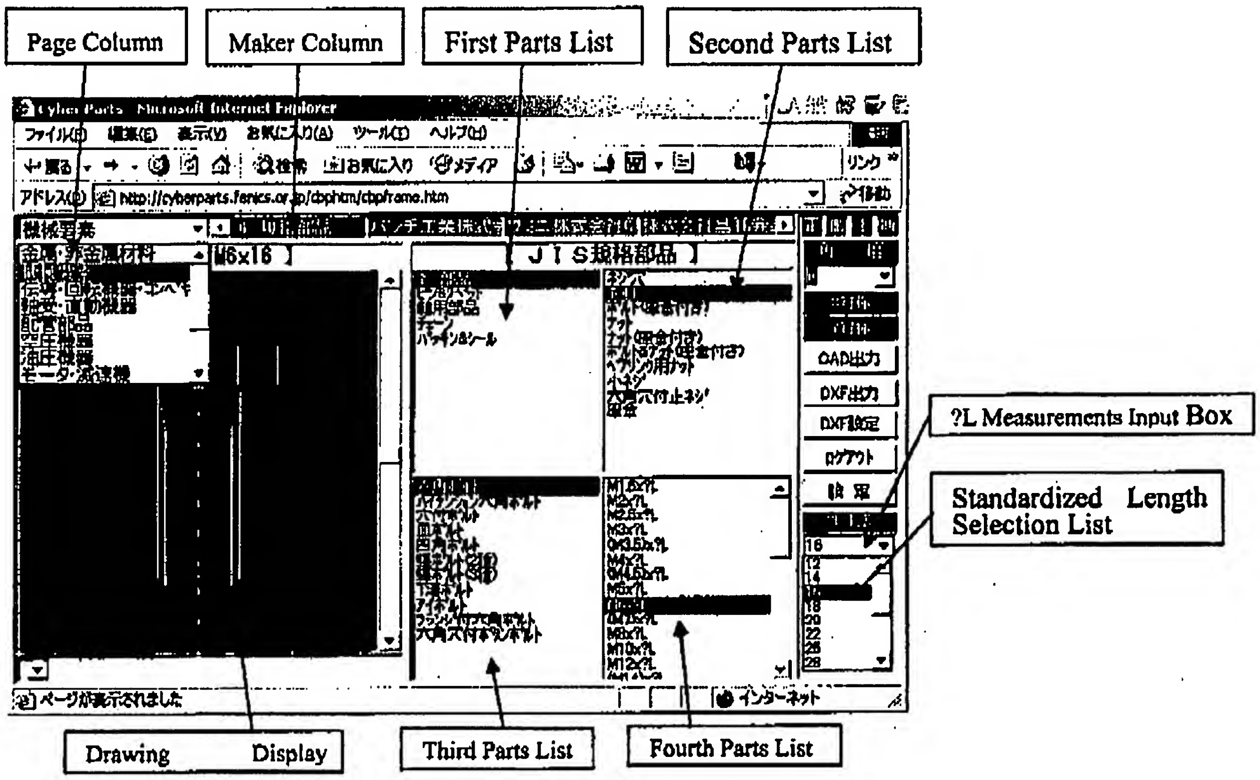


Fig1 FUJITSU Ltd.
CyberParts Parts Searching Screen

The following is an explanation in case which a part is chosen in the order of Mechanical Elements – JIS Standardized Parts – Screw Parts – Bolts – Hexagon Head Bolts:

The way to name data file is automatically decided in principle by following the criteria below:

Page column is divided into 26 kinds of A – Z. 01 – 99 are assigned to Maker Column, 01 – 99 to Fist Parts List, 01 – 99 to Second Parts List, and 01 —99 to Third Parts List. In case of hexagon head bolts, B is assigned to Mechanical Parts in Page Column because it is in the second page. 01 is assigned to JIS Standardized Pars in Maker column because it is at leftmost. 01 is assigned to Screw Parts in First Parts List because it is at top. 02 is assigned to Bolts in Second Parts List because it is at second. 01 is assigned to hexagon head bolts in Third Page List because it is at top. Therefore, the file name becomes B01010201. The extension of real number data file is set as DT; therefore, B B01010201.DT become the file name of real number data file. The variable program file is divided into the extension of PR0 (front view), PR1 (side view), PR2 (top view), and PR3 (sectional view) according to the projection side, and it becomes B010201.PR0, B010201.PR1, B010201.PR2, B010201.PR3. Normally, one part has one real number data file and four variable program files at most. The type numbers of parts which are recorded in the real number data file, B010201.DT, are displayed in Fourth Parts List. The reason why the variable program files are recorded separately for each projection side is because it becomes hindrance if there is other projection side in case that the user calls a part recorded in parts library from CAD, and pasts it directly to working CAD drawings.

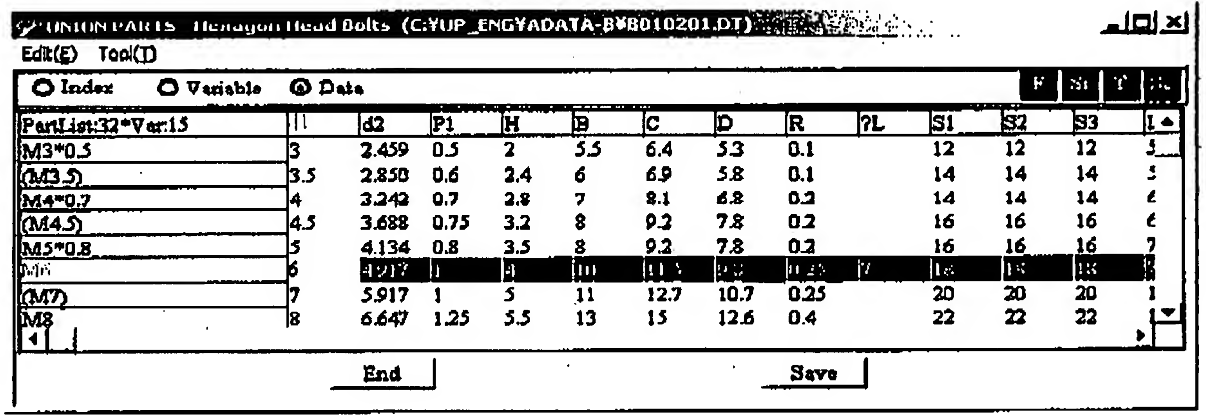
2. The Structure of Real Number Data File

The real number data files of hexagon head bolts are TXT file. If you open B01010201.DT, you will see the first number in the first line "15" which means the number of variables. In between the next commas, variables and array numbers are written. The first variable is 'd1', array number is '1' after # and the followings are the same. '012' after * shows the projection side of drawings. 0 = front view, 2 = side view, 3 = upper view, and 4 = sectional view. *012 means that there are variable programs of front view, side view, and top view for the data of B01010201. When '?' appears in front of variables, that means 'User Input Variables' which are inputted by users. '?L', which is seen below, is this variable, and the user can input the length of bolts in this case. Standardized lengths are written in the second and third lines so that the user can chose the length (See Fig2 and Fig3). This can be done by inputting standardized lengths in User Input Variable beforehand. The first letters in the second line, 'Lmin' and 'Lmax' designate the range of standardized lengths by variable. For example, in case of M6, Lmin=7, Lmax=70. Therefore, the standardized lengths are displayed in the range between 7 – 70. In case that the standardized lengths are not necessary, only Measurements Input Box is displayed. The real number data starts from the fourth line, and the type number of parts (M6) are written in the first row. Data is written from the second row in the order that the variable stands in a line. TAB divides in between the pieces of data. The Correspondence between the actual array variables written in variable program file and data in case M6 is selected is as follows:

W1=d1#1=6 W15=d2#15=4. 917 W2=P1#2=1 W3=H=4 W4=B#4=10 W5=C#5=11. 5 W6=D#6=9. 8
W7=R#7=0. 25 W8=?L#8= User's choice W9=S1#9=18 W11=S2#11=18 W12=S3#12=18 W14=Lmin#14=7
W13=Lmax#13=70 W10=k#10=1

```
15,d1#1,d2#15,P1#2,H#3,B#4,C#5,D#6,R#7,?L#8,S1#9,S2#11,S3#12,Lmin#14,Lmax#13,k#10,*012
Lmin,Lmax:5;6;7;8;9;10;11;12;14;16;18;20;22;25;28;30;32;35;38;40;45;50;55;60;65;70;75;80;85;90;95;100;105;110;
115;120;125;130;140;150;160;170;180;190;200;220;240;260;280;300;325;350;375;400
M3*0.5    3     2.459 0.5   2     5.5   6.4   5.3   0.1         12    12    12    5     32    0.6
(M3.5)    3.5   2.850 0.6   2.4   6     6.9   5.8   0.1         14    14    14    5     32    0.6
M4*0.7    4     3.242 0.7   2.8   7     8.1   6.8   0.2         14    14    14    6     40    0.8
(M4.5)    4.5   3.688 0.75  3.2   8     9.2   7.8   0.2         16    16    16    6     40    0.8
M5*0.8    5     4.134 0.8   3.5   8     9.2   7.8   0.2         16    16    16    7     50    0.9
M6   6    4.917 1     4     10    11.5  9.8   0.25        18    18    18    7     70    1
(M7) 7    5.917 1     5     11    12.7  10.7  0.25        20    20    20    11    100   1
M8   8    6.647 1.25  5.5   13    15    12.6  0.4         22    22    22    11    100   1.2
M10  10   8.376 1.5   7     17    19.6  16.5  0.4         26    26    26    14    100   1.2
```

Fig 2 Example of Real Number Data File (B01010201.DT)


UNION PARTS Hexagon Head Bolts (C:¥UP_ENG¥ADATA-B¥B010201.DT)
Edit(E) Tool(T)
Index
Variable
Data
End
Save

PartList:32*Var:15	d1	d2	P1	H	B	C	D	R	?L	S1	S2	S3	L
M3*0.5	3	2.459	0.5	2	5.5	6.4	5.3	0.1		12	12	12	5
(M3.5)	3.5	2.850	0.6	2.4	6	6.9	5.8	0.1		14	14	14	5
M4*0.7	4	3.242	0.7	2.8	7	8.1	6.8	0.2		14	14	14	6
(M4.5)	4.5	3.688	0.75	3.2	8	9.2	7.8	0.2		16	16	16	6
M5*0.8	5	4.134	0.8	3.5	8	9.2	7.8	0.2		16	16	16	7
M6	6	4.917	1	4	10	11.5	9.8	0.25	7	18	18	18	[illegible]
(M7)	7	5.917	1	5	11	12.7	10.7	0.25		20	20	20	1
M8	8	6.647	1.25	5.5	13	15	12.6	0.4		22	22	22	1

Fig 3 Embodiment of Data Edit Program

Fig 3 is an embodiment of data edit program, which was made to create real number data files to embody the present invention. It is in a state that the real number data file (B01010201.DT) was called in and displayed. Each variable character string is displayed in the first line by the same order of the first line of the real number data file (B01010201.DT). F S T Sc of upper right shows that this real number data file has the variable program files of front view, side view and top view. Sectional view box (Sc) is in blue color and this means that there is no program file of sectional view. F box is in red color and this means that front view is currently selected. Data is displayed from the second line, and the type numbers of parts are displayed in the first row. Data is displayed exactly in the same order as the fourth line of the real number data file (B01010201.Dt) from the second row.

3. The Structure of Variable Program Files

As mentioned before, in case that a type number of M6 is selected in the Fourth Parts List of Fig 1, a data of drawing is created after the operation which is written in the variable expression program file of Fig 4 is executed as following order:

1. A Data is read from the real number data file, and the data is substituted for each array variable (W1 – W15). Actually, the following numerical values are substituted for array variables which correspond to the character string written in the variable expression program files of Fig 4. The variable character string which corresponds to array variables are written in the inside of brackets for convenience. The numerical values whom array variable 'W' possesses are double precision real numbers and have no relationship with the character string in the blackest.
2. The variable program file is as same as JavaScript and VBScript. The Variable program file is a text file, and the variable program file itself does not do operation. If the variable program file is called into memory, the analysis of the character string will get started from the first line in the embodiment of the present invention. The variable program files are divided by commas and the numeral character string which is written until first comma is line number. The numeral character string until the next comma is the command number of each line. Each command number is 0=calculation, 1=segment, 2=dot, 3=circle, 4=circular arc, 5=character, 6=C square, 7=R square, 8=copy, 9=arrangement, 10=condition. The character string after that line is analyzed according to this command number. It is programmed to give the coordinates of the segments of the drawings in double precision real number by taking the following procedure: If there are the operational signs such as ^, *, /, +, −, =, etc. and the functional character strings such as SIN( ), COS( ), TAN( ), ATN( ), ABS ( ), SQR ( ), etc., the character strings of the variable program files will be analyzed according to the signs and arrays, and then those operations and functions are executed.
3. The vector drawings are displayed in the Drawing Display Column according to the given coordinate data.

```
1,10,2,28,,-1,W8<=W13,,,,,
2,0,0,0,0,-1,E1=W4*2/3^0.5,H1=(E1/2-W6/2)*TAN(P/6),R1=(H1^2+E1^2/64)/(2*H1),R2=(H1^2+E1^2/16)/(2*H1),S0=W9,
3,10,4,4,0,-1,W8>=130ANDW8<=200,S0=W11,,,,
4,10,5,5,0,-1,W8>=220,S0=W12,,,,
5,1,0,10,0,-1,E1/2,0,-E1/2,0,,
6,1,0,10,0,-1,E1/2,0,E1/2,W3-H1,,
7,1,0,10,0,-1,E1/4,0,E1/4,W3-H1,,
8,4,0,10,0,-1,3/8*E1,W3-R1,R1,90-ATN(E1/8/(R1-H1))*180/P,90+ATN(E1/8/(R1-H1))*180/P,
9,4,0,10,0,-1,0,W3-R2,R2,90-ATN(E1/4/(R2-H1))*180/P,90+ATN(E1/4/(R2-H1))*180/P,
10,1,0,10,0,-1,-3/8*E1,W3,3/8*E1,W3,,
11,8,1,1,1,-1,0,0,0,6,8,
12,4,0,10,0,-1,W1/2+W7,-W7,W7,90,180,
13,1,0,10,0,-1,-W1/2,-W7,-W1/2,-ABS(W8-W10),,
14,1,0,10,0,-1,W1/2,-ABS(W8-W10),W1/2-W10,-W8,,
15,1,0,10,0,-1,W1/2-W10,-W8,-W1/2+W10,-W8,,
16,8,1,1,1,-1,0,0,0,12,14,
17,1,0,10,0,-1,W1/2,-ABS(W8-W10),-W1/2,-ABS(W8-W10),,
18,10,19,23,0,-1,W8-S0-2*W2>0,,,,
19,1,0,10,0,-1,W1/2,-W8+S0,-W1/2,-W8+S0,,
20,1,0,11,0,-1,W1/2,-W8+S0+2*W2,W15/2,-W8+S0,,
21,1,0,11,0,-1,W15/2,-W8+S0,W15/2,-W8+W10-(W1-W15)/2,,
```

Fig 3 A Example of Variable Program File (B01010201.PRO)

4. The Way To Analyze Variable Character string

I would like to explain how the character strings are actually analyzed in detail taking the second line of the variable program file of above Fig 4 as an example.

(1) Fig 4 The contents of variable program file (B01010201) of each line are called into VALU property of the array character string variable, and line numbers are attached to array numbers. PRWORD(i) in the account of the right is an example of array character string variable. i in the brackets is the array number.	PRWORD(1).VALU= 1,10,2,28,,-1,W8<=W13,.... PRWORD(2).VALU= 2,0,0,0,0,-1,E1=W4*2/3^0.5, H1=(E1/2-W6/2)*TAN(P/6),R1=(H1^2+E1^2/64)/(2*H1), R2=(H1^2+E1^2/16)/(2*H1),S0=W9. PRWORD(3).VALU= 3 ,10,4,4,0,-1,W8>=130ANDW8<=200,S0=W11,...
(2) The array Number 2 is designated in case that the second line is analyzed.	i=2

(3) The character string in between commas are extracted from the character string of PRWORD(2).VALU, and each of them are stored in two dimension array variables. The extracted character string is stored in the two dimension array variables, SELECT(m,l).VALU in the right hand side example.	SELECT(2,1).VALUE=2
	SELECT(2,2).VALUE=0
	SELECT(2,3).VALUE=0
	SELECT(2,4).VALUE=0
	SELECT(2,5).VALUE=0
	SELECT(2,6).VALUE=-1
	SELECT(2,7).VALUE= " E1=W4*2/3^0.5"
	SELECT(2,8).VALUE= " H1=(E1/2-W6/2)*TAN(P/6)"
	SELECT(2,9).VALUE= " R1=(H1^2+E1^2/64)/(2*H1)"
	SELECT(2,10).VALUE= " R2=(H1^2+E1^2/16)/(2*H1),"
	SELECT(2,11).VALUE= "S0=W9"

(4) SELECT(2,2).VALU=0; therefore, command No=0, and this line is interpreted as calculation command. In this case, the parts that the character string analysis is necessary is from SELECT(2,7).VALU to SELECT(2,11). The analysis starts from SELECT(2,7).VALU.	SELECT(2,7).VALUE= " E1=W4*2/3^0.5"
(5) The character string after '=' is extracted and stored in data(1).value. '=' flag is substituted for data(1).property1. In case of '=', the substitution is data(1).property2='E1'	data(1).value= "W4*2/3^0.5" data(1).property1= "= " data(1).property2= "E1" data(1).index= "SELECT(2,7)"
(6) The inside of the parentheses is found from the contents of data(1).property1, and the characters in between the parentheses are tried to be extracted; however, operational sign character string is not found at all in between the parentheses. Therefore, it is skipped and the next operational sign character string is searched.	
(7) +, — are tried to be found from the contents of data(1).value, and the character string in between them is tried to be extracted; however, there are no operational sign character string such as + and —. Therefore, it is skipped and the next operational sign character string is searched.	
(8) * and / are found from the contents of data(1).value, and the character string in between them is substituted for data(2).value. The contents of character string before W4 are substituted for data(2).property1. In this case, null is substituted because there is no character string before W4. The character string after W4, * is substituted for data(2).property2. The name of array variable, data(1), which became a search object is substituted for data(2).index. The above procedure is repeated until the character string comes not to be found with adding more array index.	data(2).value= " W4" data(2).property1= " " data(2).property2= "*" data(2).index= "data(1)" data(3).value= " 2" data(3).property1= "*" data(3).property2= "/" data(3).index= "data(1)" data(4).value= " 3^0.5" data(4).property1= "/" data(4).property2= " " data(4).index= "data(1)"

(9) ˆ is found from the contents in between data(2).value – data(4).value, and the character string between them is substituted for data(5).value. The contents of operational character before '3' is substituted for data(7).property1. In this case, there is no character string before '3'; therefore, null is substituted. The content of operational character after '3', ' ˆ ' is substituted for data(5).peoperty2. The name of array variable, data(4) is substituted for data(5).index which became the search object. The above procedure is repeated until the character string comes not to be found with adding more array index.	data (5). value= " 3" data (6). property1= " " data (5). property2= "ˆ" data (5). index= "data(4)" data (6). value= " 0.5" data (6). property1= "ˆ" data (6). property2= " " data (6). index= "data(4)"
(10) Since the character string between each operator was extracted in the late turn of the execution order of an operator to the character string of the contents stored in DATA(1), operation is performed in the early turn of an execution order this time. Data(5).value ^ Data(6).value is executed and the result is substituted for keisan(1).value.	keisan(1).value=data(5).valueˆ data(6).value
(11) The contents of keisan(1).value are substituted for the array variable, data(4).value that became the searching object, and the contents of (8) are changed like the account of the right	data (2). value= " W4" data (2). property1= " " data (2). property2= "*" data (2). index= "data(1)" data (3). value= " 2" data (3). property1= "*" data (3). property2= "/" data (3). index= "data(1)" data (4). value= keisan(1). value data (4). property1= "/" data (4). property2= " " data (4). index= "data(1)"
(12) Like (10), the operation of *, and / is performed like the account of the right.	keisan(2). value=data(2). value*data(3). value/data(5). value
(13) The contents of keisan(2).value are substituted for the array variable, data(1).value that became the searching object, and the contents of (5) are changed like the account of the right.	data(1). value= keisan(2). value data(1). property1= "= " data(1). property2= " " data(1). index= "SELECT(2, 7)"

(14) The contents of data(1) are substituted for SELECT(2, 7).VALUE that became the searching object at data(1), and data(1).propaty1 has '=' flag; therefore, data(1).value is substituted for the contents of data(1).propaty2, E1. Then the calculations of E1 and SELECT(2, 7).VALUE are completed.	SELECT(2, 7). VALUE= data(1). value
	E1= data(1). value

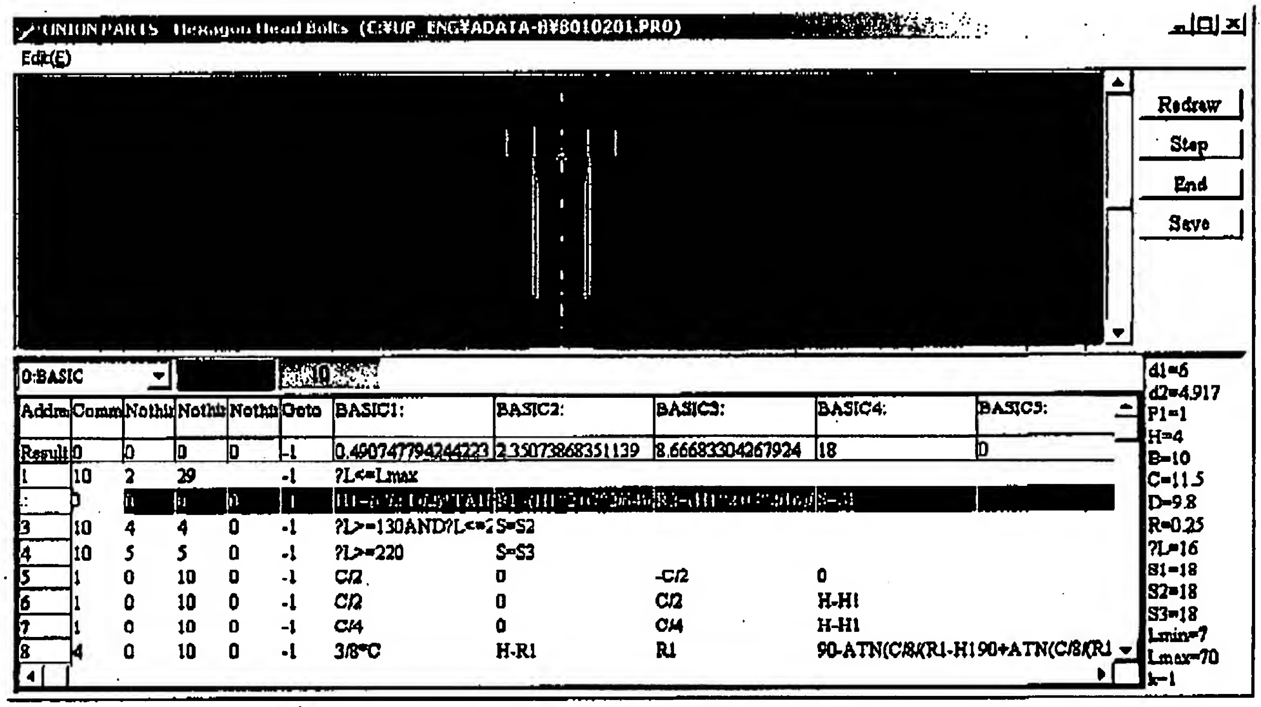
(15) Equally, SELECT(2, 8).VALUE is calculated. Data array variable is initialized. Kakko array variable is initialized. Keisan array variable is initialized.	SELECT(2, 8). VALUE= "H1=(E1/2-W6/2)*TAN(P/6)"
(16) The character string after '=' is extracted and stored in data(1).value. '=' flag is substituted for data(1).property1. In case of '=' flag, data(1).property2='H1' is substituted.	data(1). value= "(E1/2-W6/2)*TAN(P/6)" data(1). property1= "= " data(1). property2= "H1" data(1). index= "SELECT(2, 8)"
(17) ( ) is found from the contents of data(1).value, and the character string between them is substituted for the parenthesis object, kakko(1).value. The level of the parentheses is entered in kakko(1).property1. For example, the out side parentheses of (( )) become level 2. The attribute that the parentheses have is entered in kakko(1).property2. For example, in case of SIN(), it becomes SIN. In case of the account of the right, there is no attribute, so null is entered.	kakko(1). value= "E1/2-W6/2" kakko(1). property1=1 kakko(1). property2= " " kakko(1). index= "data(1)" kakko(2). value= "P/6" kakko(2). property1=1 kakko(2). property2=TAN kakko(2). index= "data(1)"

(18) +, — is found from the contents of kakko(1).value, and the character string between them is extracted and substituted for data(2).value. The operational character in front of the character string extracted is substituted for data(2).property1. In this case, there is none, so null is entered. The operational character, '—' behind the character string extracted is substituted for data(2).property2. The Index of array variable that became the searching object is substituted for data(2).index.	data(2).value= "E1/2" data(2). property1= " " data(2). Property2= "—" data(2).index= "kakko(1)" data(3).value= "W6/2" data(3). property1= "—" data(3). Property2= " " data(3).index= "kakko(1)"
(19) Equally, the character string of *, / is searched, the character string between them is extracted, and stored in each array variable like the account of the right. Thus, it repeats until each operator is no longer found.	data(4).value= "P" data(4). property1= " " data(4). Property2= "/" data(4).index= "kakko(2)" data(5).value= "6" data(5). property1= "/" data(5). Property2= " " data(5).index= "kakko(2)" data(6).value= "E1" data(6). property1= " " data(6). Property2= "/" data(6).index= "data(2)" data(7).value= "2" data(7). property1= "/" data(7). Property2= " " data(7).index= "data(2)" data(8).value= "W6" data(8). property1= " " data(8). Property2= "/" data(8).index= "data(3)" data(9).value= "2" data(9). property1= "/" data(9). Property2= " " data(9).index= "data(3)"
(20) Since the character string between each operator was extracted in the late turn of the execution order of an operator from the character string of the contents stored in DATA(1), an operation is performed in the early turn of an execution order this time, and the each result is substituted for keisan(1).value – keisan(3).value.	Keisan(1).value= data(4).value/ data(5).value Keisan(2).value= data(6).value/ data(7).value Keisan(3).value= data(8).value/ data(9).value
(21) From the contents of (20), the each array variable of kakko(2), data(2), data(3) is converted like the account of the right.	data(2).value= Keisan(2).value data(2). property1= " " data(2). Property2= "—" data(2).index= "kakko(1)" data(3).value= Keisan(3).value data(3). property1= "—" data(3). Property2= " " data(3).index= "kakko(1)"
(22) From data (2) and data (3), — operation is executed and the result is stored in keisan(5)	Keisan(5)= data(2).value—data(3).value
(23) Because of (20) and (22), the value of kakko(1) and kakko(2) become as the account of the right. The value of kakko(2).value becomes the one that the tangent operation is automatically done because kakko(2) is with TAN attribute. The calculation in a parenthesis was completed above.	Kakko(1).value=keisan(5).value kakko(1).property1=1 kakko(1).property2= " " kakko(1). index= "data(1)" Kakko(2).value= Keisan(1).value kakko(2).property1=1 kakko(2).property2=TAN kakko(2).index= "data(1)"
(24) If a result is substituted for data(1), it will become like the account of the right. Since there came to be no parenthesis, the search of other operators is performed equally.	data(1).value= "Kakko(1).value * Kakko(2).value" data(1).property1= "= " data(1).property2= "H1" data(1).index= "SELECT(2,8)"
(25) * operator is found, and the contents of the right are executed.	data(10).value = Kakko(1).value data(10).property1= " " data(10).property2= "*" data(10).index= "data(1)" data(11).value = Kakko(2).value data(11).property1= "*" data(11).property2= " " data(11).index= "data(1)"

(26) The operation of data(10) and data(11) is executed and the result is stored in keisan(6).	Keisan(6)= Kakko(1).value* Kakko(2).value
(27) The result of (26) is substituted for data(1).	data(1).value= "keisan(6)" data(1).property1= "= " data(1).property2= "H1" data(1).index= "SELECT(2,8)"

(28) The contents of data(1) are substituted for SELECT (2, 8).VALUE that became the searching object at data(1), and data(1).propaty1 has '=' flag; therefore, data(1).value is substituted for the contents of data(1).propaty2, H1. Then the calculations of H1 and SELECT(2, 8).VALUE are completed.	SELECT(2,8).VALUE= data(1).value
	H1= data(1).value

As explained above, the character analysis is carried out one by one for the contents written in variable program files. If there are any operatinal character, the character string before and behind of them is extracted in the late turn of the execution order of operation, and the operational character is kept in the property of the array variable. The name of array variables and number which became a searching object are stored in *(i).index of the array variable. Thus, the variable program files are all comprehended according to the format, the operation as the contents currently written is performed, and the vector data of drawings is automatically generated.


UNION PARTS Hexagon Head Bolts (C:¥UP_ENG¥ADATA-B¥B010201.PRO)
Edit(E)
Redraw
Step
End
Save
0:BASIC

Addre	Comm	Nothi	Nothi	Nothi	Goto	BASIC1:	BASIC2:	BASIC3:	BASIC4:	BASIC5:
Result	0	0	0	0	-1	0.490747794244223	2.35073868351139	8.66683304267924	18	0
1	10	2	29		-1	?L<=Lmax				
2	0	0	0	0	1	[illegible]	[illegible]	[illegible]	[illegible]	
3	10	4	4	0	-1	?L>=130AND?L<=2	S=S2			
4	10	5	5	0	-1	?L>=220	S=S3			
5	1	0	10	0	-1	C/2	0	-C/2	0	
6	1	0	10	0	-1	C/2	0	C/2	H-H1	
7	1	0	10	0	-1	C/4	0	C/4	H-H1	
8	4	0	10	0	-1	3/8*C	H-R1	R1	90-ATN(C/8/(R1-H1 90+ATN(C/8/(R1	

d1=6
d2=4.917
P1=1
H=4
B=10
C=11.5
D=9.8
R=0.25
?L=16
S1=18
S2=18
S3=18
Lmin=7
Lmax=70
k=1

Fig 5

Fig 5 is an embodiment of a parametric drawing tool which was developed to create parametric drawing of the present invention. The real number data is called in from the real number data file (B01010201.DT), data is substituted for the array variable, the variable program file (B01010201.PRO) is analyzed as mentioned before, the operation is executed and the coordinates data of the drawing element is created form the result, the vector drawing is displayed in the upper part, and the contents which were called in form Fig 4 Variable Program Files (B01010201) are displayed in a table form in the lower part. For example, in case that the address number of above drawing is of fifth line, the array variable corresponding to the red part of the drawing element is displayed in a table form, and the each element of the drawing and the each line of the table in the lower part correspond one by one.

5. The Explanation of Each Command

I will explain the detail of each command which was explained in ② of '3.The Structure of Variable Program Files' based on the parametric drawing tool of Fig 5. In fig 5, the first row of the variable table in the lower part shows line number (address), and the second row shows command number. The contents after the third row depend on each command in the second row, and it is as follows:

(0) Calculation Command
Calculation command is able to make complicated variable expression substituted for user variables beforehand, and the variable can be used after the substitution. Therefore, it dramatically simplifies coordinates designation of an element.

3rd row	4th row	5th row	6th row	7th row	8th row	9th row	10th row	11th row
None	None	None	GOTO	Calculation1	Calculation2	Calculation3	Calculation4	Calculation5

The calculation result of the variable expression written in calculation 1 column is reflected in calculation 2 column. It is always reflected in a lower streamside, and is not reflected in an upper streamside. In case that coordinates calculation becomes complicated variable expression, coordinates designation of each element can be easily done by substituting the calculation result for the user variable beforehand.

(1) Segment Command
Line Segment command draws the segment from a starting point of coordinates position to an end point of coordinates position in designated line color and type. The line color number is set in the 4th row by designating the line color from the line color column. The line type number is set in the 3rd row by designating the line type from the line type column. Numerical value can directly be inputted for line color and type. The designating columns of the starting point and end point of coordinates is from 7th row to 10th row as shown in the following table. Coordinates are designated by the variable expression or the numerical value.

3rd row	4th row	5th row	6th row	7th row	8th row	9th row	10th row	11th row
Line Type No.	Line Color No.	None	GOTO	Coordinates starting point X1	Coordinates starting point Y1	Coordinates end point X2	Coordinates end point Y2	None

(2) Dot Command
Dot Command does only color designation and coordinates designation. It sets line color in the 4th row, X coordinate in the 7th row and Y coordinate in the 8th row.

3rd row	4th row	5th row	6th row	7th row	8th row	9th row	10th row	11th row
None	Line Color No.	None	GOTO	Coordinates X1	Coordinates Y1	None	None	None

(3) Circle Command
Circle command draws circles at center coordinate in designated radius, line color and line type. The line color number is set in the 4th row by designating the line color from the line color column. The line type number is set in the 3rd row by designating the line type from the line type column. Center coordinate and radius are designated by the variable expression or the numerical value from 7th row to 9th row as shown in the following table.

3rd row	4th row	5th row	6th row	7th row	8th row	9th row	10th row	11th row
Line Type No.	Line Color No.	None	GOTO	Center Coordinates X1	Center Coordinates Y1	Radius R	None	None

(4) Circular Arc Command
Circular arc command draw arcs at center coordinate in designated radius, line color and line type from starting angle to end angle. The designated unit of starting angle and end angle is degree. The line color number is set in the 4th row by designating the line color from the line color column.
The line type number is set in the 3rd row by designating the line type from the line type column. Center coordinate, radius, starting angle, and end angle are designated by the variable expression or the numerical value from 7th row to 11th row as shown in the following table.

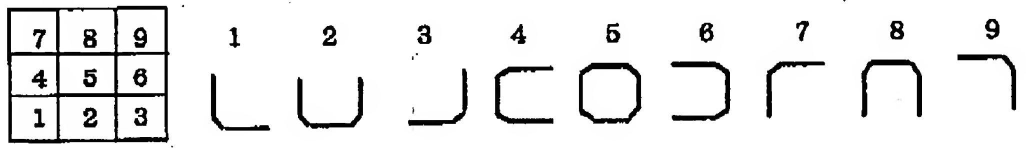
3rd row	4th row	5th row	6th row	7th row	8th row	9th row	10th row	11th row	12th row
Line Type No.	Line Color No.	None	GOTO	Center Coordinates X1	Center Coordinates Y1	Radius R	None	None	None

(5) Character Command
Character command creates character by setting that the designated coordinates are at left end in designated character color, character angle, character height, and character width. Character angle is designated in degree. Character height and width are designated in mm. The designated position of each column is as follows.

3rd row	4th row	5th row	6th row	7th row	8th row	9th row	10th row	11th row	12th row
None	Character Color	None	GOTO	Character Coordinates X1	Character Coordinates Y1	Character height	Character Angle	Character Data	None

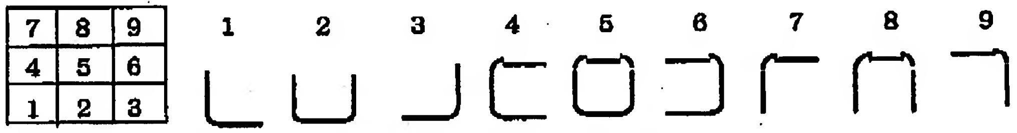
(6) C Quadrangle Command
C quadrangle command draws the quadrangle, L-shaped type and a U-shaped type of which C chamfering was done. The angle position of the fifth row has the same relation as layout of the number key of a keyboard, and is shown as follows. If 0 is inputted in the 11th row, chamfering is not carried out.

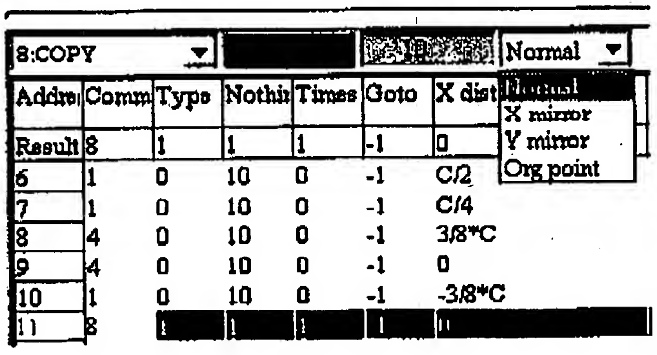
3rd row	4th row	5th row	6th row	7th row	8th row	9th row	10th row	11th row
Line Type No.	Line Color No.	Angle Position	GOTO	Opposite Angle Coordinates X1	Opposite Angle Coordinates Y1	Opposite Angle Coordinates X2	Opposite Angle Coordinates Y2	Chamfer Dimension



(7) R Quadrangle Command
R quadrangle command draws the quadrangle, L-shaped type and a U-shaped type of which R chamfering was done. The angle position of the fifth row has the same relation as layout of the number key of a keyboard, and is shown as follows. If 0 is inputted in the 11th row, chamfering is not carried out.

3rd row	4th row	5th row	6th row	7th row	8th row	9th row	10th row	11th row
Line Type No.	Line Color No.	Angle Position	GOTO	Opposite Angle Coordinates X1	Opposite Angle Coordinates Y1	Opposite Angle Coordinates X2	Opposite Angle Coordinates Y2	Chamfer Dimension




8:COPY
Normal
Normal
X mirror
Y mirror
Org point

Addre	Comm	Type	Nothi	Times	Goto	X dist
Result	8	1	1	1	-1	0
6	1	0	10	0	-1	C/2
7	1	0	10	0	-1	C/4
8	4	0	10	0	-1	3/8*C
9	4	0	10	0	-1	0
10	1	0	10	0	-1	-3/8*C
11	8	1	1	1	1	0

(8) Copy Command

Copy command copies the element from the start address of 11th and 10th row to the end address in form of the contents of the copy type of the 3rd row. If a copy command is selected, 'symmetrical type column' of the account of the left is displayed at the next of line color column. If a symmetrical type is selected, a type number is set in the type column of the 3rd row. The types and functions of copies are as follows:

0 Normal	== An parallel copy is carried out at the distance designated in the 'XY direction distance column' of 7th and 8th row. If the number of times is set to 'the continuous-number-of-times column' of the 5th row, it will be copied in the designated pitch only the number of times of designation in the 'XY direction distance column'. If an angle is set to the 'rotation angle column' of the 9th row, it will be rotated and copied at the angle. A unit is designated with a degree.
1 X mirror	== It copies symmetrically with respect to the vertical axis which passes through the position designated in the 'X direction distance column' of the 7th row. If the number of times is set to 'the continuous-number-of-times column' of the 5th row, it will be copied symmetrically with the respect to the vertical axis in the designated pitch only the number of times of designation in the 'X direction distance column'. In this case, the numerical value of 'Y direction distance column' is ignored. If an angle is set to the 'rotation angle column' of the 9th row, it will be rotated and copied at the angle. A unit is designated with a degree.
2 Y mirror	== It copies symmetrically with respect to the X axis which passes through the position designated in the 'Y direction distance column' of the 8th row. If the number of times is set to 'the continuous-number-of-times column' of the 5th row, it will be copied symmetrically with the respect to the horizontal axis in the designated pitch only the number of times of designation in the 'Y direction distance column'. In this case, the numerical value of 'X direction distance column' is ignored. If an angle is set to the 'rotation angle column' of the 9th row, it will be rotated and copied at the angle. A unit is designated with a degree.
3 Origin	== It copies symmetrically with respect to the origin having the origin (0, 0) as center. Parallel translation is carried out to the distance designated and copied if the numerical value is set to the 'XY direction distance column' designated in the 7th row and the 8th row. If the number of times is set to 'the continuous-number-of-times column' of the 5th row, it will be copied in the designated pitch only the number of times of designation in the 'XY direction distance column'. If an angle is set to the 'rotation angle column' of the 9th row, it will be rotated and copied at the angle. A unit is designated with a degree.

3rd row	4th row	5th row	6th row	7th row	8th row	9th row	10th row	11th row
Copy Type	None	Continuous number of times	GOTO	X Direction Distance	Y Direction Distance	Rotation Angle Column'	Start Address	End address

(9) Placement Command

The placement command has not been presently registered.

(10) Condition Command

Condition command is used in order to change the flow of a program or to carry out function calculation by following the condition.

The condition column of 7th row is used when comparing two variable expressions and numerical values or checking two or more conditions using a relational operator (><=) and a logical operator (AND, OR), and a comparison result is obtained in truth and false.

Example

A>B	When A is bigger than B, "truth". When other time, "false".
A=B	When A and B are equal, "truth". When other time, "false".
A<B and B<C	When A is smaller than B and B is smaller than C, "truth". When other time, "false".

The 3rd and 4th rows are used in order to change the flow of program according to the comparison result. When an address is designated in the 3rd row and the result is truth, it jumps to the address. When an address is designated in the 4th row and the result is false, it jumps to the address.

3rd row	4th row	5th row	6th row	7th row	8th row	9th row	10th row	11th row
GOTO True	GOTO False	None	None	Condition	Function 1 True	Function 2 True	Function 1 False	Function 2 False

The 8th to 11th rows are used when calculating the variable formula inputted according to the comparison result. The 8th and 9th rows calculate when the comparison result is truth. The 10th and 11th rows calculate when the comparison result is false.

I explained how the vector data is created by each command of a variable program files above.

Next, I describe the system variables, system functions, and arithmetic operators currently used in the embodiment of the present invention.

If any character string of this system variables, system functions, arithmetic operators, the above-mentioned relational operators, and logical operators are discovered in the variable program, the embodiment of the present invention is programmed to perform the operation that the discovered character string means based on it using the numerical values that were substituted for each array variables (W1, W2...) and to calculate the vector data of the drawing.

6. The Explanation of the System Variable of the Embodiment

USER:	Variable for real data drawings
P:	System variable (the ratio of the circumference of a circle to its diameter)
PI:	System variable (the ratio of the circumference of a circle to its diameter)
Wi:	System variable (the variable for arrays)
WWi:	System variable (the variable for arrays)
UCM:	System variable (the variable to chamfer for C quadrangle and R Quadrangle)
UK1:	System variable (the start angle of a circular arc)
UK2:	System variable (the end angle of a circular arc)
UR:	System variable (the radius of circle and circular arc)
UA:	System variable (the X coordinate of the starting point of line segments, the X coordinate of dots and the X coordinate of centers of circles and circular arcs)
UB:	System variable (the Y coordinate of the starting point of line segments, the Y coordinate of dots and the Y coordinate of centers of circles and circular arcs)
UX:	System variable (the X coordinate of the end point of line segments)
UY:	System variable (the Y coordinate of the end point of line segments)
@:	Program variable (If this variable is in a data file, the character string of this variable data will be added to a program file name, and it will be called out to it. It is used when the user want to create the program of different form from the same data)
?:	Numerical value input variable (If this character is used for the head of a variable, length can be inputted each time at the time of part selection)

7. The Explanation of the Functions of the Embodiment

1. SIN( ), sin( ) --- sine
   The sine of the expression in parentheses is returned. The unit of expression is radian.
2. COS( ), cos( ) --- cosine
   The cosine of the expression in parentheses is returned. The unit of expression is radian.
3. TAN( ), tan( ) --- tangent
   The tangent of the expression in parentheses is returned. The unit of expression is radian.
4. ATN( ). atn( ) --- arc tangent
   The arc tangent of the expression in parentheses is returned by radian. The value that can be acquired is from $-\pi/2$ to $\pi/2$.
5. ABS( ), abs( ) --- absolute value
   The absolute value of the expression in parentheses is returned.
6. SQR( ), sqr( ) --- square root
   The square root of the value designated by the expression in parentheses is returned.

8. The Arithmetic Operator of the Embodiment

Operator	Operation	Example	The execution order of the operation
^	Exponent operation	X^Y	1
*	Multiply operation	X*Y	2
/	Division operation	X/Y	2
+	Addition operation	X+Y	3
-	Subtraction operation	X-Y	3
=	Substitution operation	X=Y	4

In case of changing the execution order of operation, a parenthesis ( ) operator is used. The operator in parentheses is performed ahead of other operators. It follows in order of the usual execution in parentheses. A substitution operator is substituted for the variable of the left-hand side after all operations written in the right-hand side are performed.

9. The Handling of DXF Data and CAD Data

The data form of the present invention such as the vector data of DXF data and CAD data can be treated on a par with drawings of parametric data. Since CAD data and DXF data are real number vector data, this replaces CAD data and DXF data with the real number instead of the variable of the above-mentioned variable expression program file, and transforms and stores them in the following files for the embodiment of the present invention. Since the data format is made the same, it can be run by the same program. The example of left-hand side is the one that common CAD data and DXF data are transformed into the format of the embodiment of the present invention and stored by placing real number data instead of each array variable of a variable program file.

```
1, 1, 1, 15, , , -50. 5, -75, -50. 5, -85, ,
2, 1, 1, 16, , , -59. 5, -75, -59. 5, -85, ,
3, 1, 1, 15, , , 50. 5, -75, 50. 5, -85, ,
4, 1, 1, 16, , , 59. 5, -75, 59. 5, -85, ,
5, 1, 0, 15, , , -8. 149811, -50. 499981, -8. 149811, -54. 499981, ,
6, 1, 0, 15, , , -11. 180206, -54. 499981, -8. 149811, -54. 499981, ,
7, 1, 0, 15, , , -9. 970611, -49, -4. 899811, -49, ,
8, 4, 0, 15, , , -9. 399811, -51. 73698, 6. 5, 112. 352489, -144. 6254
88,
9, 4, 0, 15, , , -8. 899811, -52. 05048, 3. 5, 135. 546082, -139. 4709
78,
10, 4, 0, 15, , , -11. 180206, -54, 0. 5, -139. 469788, -90,
```